Electronic Version
Stylesheet Version v1.1.1

Description

Address Error Detection by Merging a Polynomial-Based CRC Code of Address bits with Two Nibbles of Data or Data ECC Bits

BACKGROUND OF INVENTION

[0001] This invention relates to error detection and correction, and more particularly to address error detection merged with data error detection and correction.

[0002] Digital memories are susceptible to errors caused by a variety of sources. Cosmic radiation can flip the state of individual memory cells. Pattern-sensitive capacitive coupling, noise, and hardware failures such as shorts can occur, causing multiple bits to be read incorrectly. Sometimes entire memory chips can fail. When a memory contains several memory chips, such as on a memory module, a one-chip failure may produce a multi-bit error, such as a 4-bit error in a 72-bit memory word.

[0003] Additional bits are often included in the memory for storing an error-correction code (ECC). These additional ECC bits can be used to detect an error in the data bits being read, and can sometimes be used to correct those errors. Typically, a code is selected such that the data is unmodified. All error detection and correction is done by comparing the check bits read against the correct check bits for that data. Such a code is considered in "systematic form".

[0004] Various codes can be used for the ECC bits, such as the well known Hamming codes. A class of codes known as Single-byte Error-Correcting / Double-byte Error-Detecting (SbEC/DbED) codes can correct any number of errors within a "byte" and detect pairs of such errors. The "byte" may be a length other than 8 bits. For example, a S4EC/D4ED code can correct 4-bit (nibble) errors, and detect but not correct 8-bit (2 nibble) errors. These codes are especially useful since they can detect double-chip errors where all 4 bits output by a two different memory chips are faulty. Single-chip errors can be corrected.

[0005] A SbEC/DbED code with 3*b check bits can be used with up to b*(2**b + 2) total bits (data + check). These are known as Reed-Solomon SbEC-DbED codes. When b=4, only a relatively small a number of data bits can be used

(60). To increase the allowed number of data bits, 4*b check bits are typically used, such as 128 data bits with 16 check bits. The increased number of check bits allows a larger number of data bits to be used.

[0006] While such S4EC/D4ED codes are useful for protecting against failures in whole memory chips, and in the wires to and from the memory chips, failures can also occur in the address lines to one or more of the memory chips. For example, a solder connection to an address pin of one of the memory chips might start failing after some time. Many memory chips use multiplexed addresses, where the address is applied over the same address lines in two parts, a row address part and a column address part. A single solder connection can thus cause two bits of the address to be faulty. It is desirable to protect against such 2-bit address errors. Some of the memory errors may be caused by cosmic radiation. This may cause a wrong address to be read from within the memory chip. This address may be wrong in an unknown number of bits.

[0007] As memory sizes increase, more and more address bits are used. Protecting these larger addresses against errors becomes more important.

[0008] Figure 1 shows a prior-art memory with data ECC and ad-

dress parity. Write data is stored in data RAM 10, while ECC generator 16 calculates the ECC bits that correspond to the value of the data bits being written into data RAM 10. These data ECC bits are written into data ECC RAM 12 at the same write-address W_ADR as the data.

[0009] During reading, the read address R_ADR is applied to read out data from data RAM 10 and data ECC bits from data ECC RAM 12. Read ECC generator 20 re-generates an ECC value from the data being read from data RAM 10. The new ECC value from read ECC generator 20 is compared to the stored ECC bits from data ECC RAM 12 by ECC checker 24 to determine if any errors occurred in the read data. A data error can be signaled when the stored ECC does not match the re-generated ECC. Some of these data errors may be corrected by an ECC corrector (not shown).

[0010] To protect against errors in the address, the write address W_ADR is applied to parity generator 18, which generates the parity of the write address. The generated address parity is then stored in address parity RAM 14 at the write address.

[0011] During reading, the stored address parity is read from address parity RAM 14, while the parity of the read address R_ADR is generated by read parity generator 22. The gen-

erated read-address parity is compared to the stored parity from address parity RAM 14 by parity comparator 26. When the parity values mis-match, and address error is signaled. The memory read can be re-tried several times before a failure is signaled.

[0012] Figure 2 shows address parity concatenated with data ECC bits. The address parity and data ECC bits can be stored in separate RAMs, or can be concatenated and stored in the same RAM. A data word of 128 bits may need 16 data ECC bits to correct errors up to 4 bits in a nibble and to detect pairs of such errors in separate nibbles. A 32-bit address protected with a standard Hamming code would need 6 bits, allowing detection of all 1 and 2 bit errors in the address. Thus a total of 23 check bits are needed to protect against both address and data errors.

[0013] Some memories may lack a sufficient width to store all of the check bits. For example, there may only be space for 16 check bits. It may be undesirable to reduce the number of data ECC bits to fit in some address parity bits. There are trade-offs among the number of check bits and expense of the memory system, the largest multi-bit data error that can be corrected and detected, and the degree of detection of address errors. Adding additional check

bits for the address parity is often undesirable. Reducing the number of address check bits can reduce detection for multi-bit address errors. The use of multiplexed address bits causes 2-bit address errors to be as likely as 1-bit address errors in a real system.

[0014] The address parity bits could be exclusive-OR'ed (XOR'ed) into the data ECC bits. This has the advantage of not requiring additional check bits. However, if the address has a parity error, the extracted data ECC bits may not be able to correct an otherwise correctable data error. Thus some data correction ability may be lost. This happens if the address error causes an error syndrome to be created that matches the error syndrome for an otherwise correctable data error.

[0015] What is desired is a memory with data error correction and detection and address error detection. It is desirable to combine address check bits with data ECC bits.

BRIEF DESCRIPTION OF DRAWINGS

[0016] Figure 1 shows a prior-art memory with data ECC and address parity.

[0017] Figure 2 shows address parity concatenated with data ECC bits.

[0018] Figure 3 shows generation of a combined data and ad-

dress check word.

[0019] Figure 4 shows storage of the merged ECC codeword in memory.

[0020] Figure 5 shows reading and checking of data and address using a merged ECC codeword.

[0021] Figure 6 shows a CRC-code generator.

DETAILED DESCRIPTION

[0022] The present invention relates to an improvement in address error detection. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

[0023] The inventor has realized that duplicating the address check bits and redundantly merging the address check bits into two nibbles of the data ECC bits can improve ac-

curacy of data correction and address checking. Since the address check bits are merged with the data ECC bits, additional bits are not needed for storing the address check bits.

[0024] Rather than generating parity of the address, a more complex cyclical-redundancy-check (CRC) code is used. CRC codes are characterized by a generator polynomial. CRC codes have well-known benefits for increased error coverage, for a given number of check bits. The benefits include better coverage for random numbers of errors, and better coverage for errors that occur in consecutive bits (bursts).

[0025] While the generation of check bits using a CRC code in hardware and software is well known, a short description follows. CRC algorithms use modulo-2 arithmetic. Only 1's and 0's are used, and there is no borrow or carry operations in the arithmetic. Binary arithmetic additions and subtractions become simple XORs.

[0026] The algorithm treats all bit streams as Binary Polynomials. A Binary Polynomial is a polynomial with coefficients implied by the bit stream, for example X**3 + X + 1. The bit stream 101011, for example, can be represented by the polynomial X**5 + X**3 + X**1 + X**0 = X**5 + X**3 + X

+ 1. A logical left shift of i bits can be represented by multiplying the polynomial by X**i.

[0027] The check bits are generated so a concatenation of the bit stream and check bits is exactly divisible by some pre-defined generator polynomial. If the bit stream is k bits, and the check bits are n bits, the generator polynomial represents n+1 bits.

[0028] To create the check bits, the data polynomial is left shifted by n bits and divided by the generator polynomial. This is all done in modulo-2 arithmetic. The remainder polynomial implies the check bits. A hardware implementation of this is typically understood in terms of a shift register and XOR gates that take multiple cycles to execute. This operation can be unfolded and parallelized, so that it all happens in one cycle. The result is a set of XOR operations on the original bit stream. The correct set of XOR operations is implied by the CRC generator polynomial.

[0029] Address errors are not correctable using any of the redundant information in the check bits from the initial data read operation, since the wrong address was read. It is thus desirable to ensure that all detectable address errors are reported as uncorrectable data errors in the SbEC/

DbED code used. This causes a restriction on the number of address check bits that can be used. The greatest address error coverage is desired using the smallest number of address check bits. Using too large a number of address check bits merged into the SbEC/DbED code for the data may cause some number of address errors to be unreported. This detracts from the benefit of the increased number of address check bits. So there is a strong coupling between the error detection capability of the address code, the number of bits used, and the ability of the data code to correctly report all such detected address errors.

[0030] Figure 3 shows generation of a combined data and address check word. Data to be written to memory is input to data ECC generator 32. In this example 16 bytes (128 bits) of write data W_DATA are input, but other widths are contemplated. Data ECC generator 32 generates a S4EC/D4ED ECC code that can correct errors of 1-4 bits, and detect but not correct errors from two groups of 1-4 bits in the 128-bit data. Various strategies are used to generate this type of ECC code. Data ECC generator 32 outputs 16-bit data ECC codeword 36, which has four nibbles DE3, DE2, DE1, DE0.

[0031] The address to write the data to, W_ADR, is a 32-bit ad-

dress. The write address is applied to CRC-code generator 34, which uses a generator polynomial to operate on the address, which is also represented as a polynomial, to generate a 4-bit output, labeled AE, address error check bits 38. The CRC generation is performed in modulo-2 arithmetic, which causes the logic function to be a series of XOR's.

[0032] Address error check bits 38 (AE) are merged with two of the four nibbles of data ECC codeword 36. XOR gates 44 merges the 4 bits of address error check bits 38 with the lowest-order nibble DE0 of address error check bits 38 to generate merged ECC nibble XE0 of merged ECC codeword 30. XOR gates 42 redundantly merges the 4 bits of address error check bits 38 with the next-lowest-order nibble DE2 of address error check bits 38 to generate merged ECC nibble XE1 of merged ECC codeword 30.

[0033] The upper two nibbles of data ECC codeword 36 are copied to the upper two nibbles of merged ECC codeword 30. Thus merged ECC codeword 30 contains two unaltered data ECC nibbles that contain only data ECC information and two merged nibbles that contain both data ECC and address check information.

[0034] Figure 4 shows storage of the merged ECC codeword in

memory. Write data 31 is written to a location in data RAM 10 pointed to by write address W_ADR. Merged ECC codeword 30 is written to the same location pointed to by write address W_ADR, but in ECC RAM 40. Data RAM 10 and ECC RAM 40 could include some of the same memory chips, or could be in separate memory chips.

[0035] Since merged ECC codeword 30 includes the address check bits merged with the data ECC bits, no additional storage is needed for address check bits. A 16-bit wide memory can store both data ECC and address check information. Cost is reduced since additional memory is not needed to store the address check bits. Alternatively, the number of data ECC bits does not have to be reduced to make room for storing address check bits. Merged ECC codeword 30 contains two merged nibbles XE1, XE0 with both address and data ECC information, and two unmerged nibbles DE3, DE2 with unaltered data ECC information.

[0036] When data is to be read from memory, read address R_ADR is applied to data RAM 10 and ECC RAM 40, selecting a location to read from. The read data R_DATA is output from data RAM 10 as 128 data bits, and the merged ECC bits are output from ECC RAM 40 as a 16-bit read

ECC codeword E_ECC.

[0037] Figure 5 shows reading and checking of data and address using a merged ECC codeword. The read data R_DATA that was read from the data RAM is input to read-data ECC generator 52, which uses a S4EC/D4ED algorithm to re-generate the 16-bit data ECC for the stored data. The re-generated data ECC contains nibbles DE3, DE2, DE1, and DE0 and is output from read-data ECC generator 52 as read-generated data ECC codeword 56.

[0038] The read address, R_ADR, is input to read-address CRC-code generator 54, which uses the same CRC polynomial as CRC-code generator 34 (Fig. 3), except it operates on the read address rather than the write address. The 32-bit read address if compressed down to 4 address check bits by read-address CRC-code generator 54, which are outputs as read-address error check bits 58.

[0039] Four nibbles are read from ECC RAM 14 (Fig. 4) as read ECC codeword 30'. The lower 2 nibbles XE1, XE0 of read ECC codeword 30' contain merged address and data check bits. XOR gates 64 exclusive-OR's the 4-bit read-address error check bits 58 from read-address CRC-code generator 54 with the lowest nibble XE0 of read ECC codeword 30'. Since two consecutive XOR operations cancel each

other, XOR gates 64 remove the merged address check bits from nibble XE0, recovering nibble DE0, which has only data ECC information and no address check information. An XOR represents an addition or subtraction in modulo-2 arithmetic.

[0040] Likewise, the same 4 bits of read-address error check bits 58 are redundantly applied to XOR gates 62, which recover data ECC nibble DE1 from merged ECC nibble XE1. Recovered data ECC codeword 60 contains only data ECC information.

[0041] Data ECC comparator 70 compares read-generated data ECC codeword 56 to recovered data ECC codeword 60. When the two codewords match, no data error occurred, and the read data can be used. When a mis-match occurs, error correction can be attempted using recovered data ECC codeword 60. The XOR of the expected and actual ECC codeword is known as the error syndrome, or syndrome. For many ECC codes, an all-zeros syndrome indicates no error, while a non-zero syndrome indicates an error. The syndromes may be sued to attempt correction of the detected error, or the error may be determined to be detected but not correctable. An ECC engine or programmable process may be used to implement the more

complex functions of error correction.

[0042] An ECC code may be constructed to guarantee that the syndrome for each cause of a correctable error is unique with respect to all other syndromes for all correctable error cases, and that all correctable error syndromes are unique with respect to all guaranteed detectable uncorrectable error syndromes. The recovery logic does a mapping of all correctable error syndromes to a correctable error signal, as well as creation of a data bit flip vector, that toggles any and all data bits that need to be corrected. For instance, since all errors within a nibble are correctable in a S4EC/D4ED code, there are 15 error syndromes for correctable errors in data bits [3:0], which is the 16 value possibilities for 4 bits, minus 1, which represents the non-corrected case. Eight of these syndromes cause correction of data bit zero, for instance. Since there are 20 nibbles of information in a S4EC/D4ED code covering 128 data bits with 16 check bits, the total number of correctable error syndromes is 20 * 15 = 300. The error syndromes that map to correctable errors of the check bits are simply corrected by doing nothing to the 128 bits of data. This is another advantage of the systematic code.

[0043] When the two lower nibbles DE1, DE0 of read-generated

data ECC codeword 56 and recovered data ECC codeword 60 both mis-match, an address error may be detected. Since the address check bits are redundantly merged with both lower nibbles DE1, DE0, an address error causes both nibbles to mis-match, rather than only one nibble. This two nibble mismatch is guaranteed to create an error syndrome that is unique with respect to all correctable data errors, due to the S4EC/D4ED ECC code used. Address error detector 72 signals an address error when both lower nibbles mismatch in data ECC comparator 70. The read cycle can be aborted and a new read of the memory attempted. After several failed read attempts, a fatal read error can be signaled.

[0044] Since an address error may cause the wrong location in memory to be read, all the bits of data may be bad. Address errors are thus a higher priority than data errors and must be corrected before considering any data errors.

[0045] When a multi-byte read-data error occurs, it is possible that both lower nibbles XE1, XE0 falsely indicate an address error. Since any multi-byte read data error is uncorrectable with SbEC/SbED codes, the memory read can be re-executed with a new address being sent to the memory. On the second read attempt, a correct address should

be transmitted when the address error was an intermittent error, such as might be caused by radiation or noise. If the address failure is confined to a single DRAM chip, such that only 4-bits of data are wrong, the syndrome decode indicates that the data is correctable using the normal data correction mechanism. A single 1-4 bit error within an aligned 4-bit byte is correctable.

[0046] Figure 6 shows a CRC-code generator. CRC-code generator 34 can be used to generate the 4 address check bits from the write address, and from the read address. The polynomial X**4 + X + 1 is implemented by four multi-input XOR gates 82, 84, 86, 88. This is a parallelization of a linear-feedback shift-register (LFSR) implementation, but shift registers or other logic could also be substituted. Each of XOR gates 82, 84, 86, 88 may be an array of 2-input XOR gates, or may be implemented in arrayed logic or using other lower-level gates. It is commonly understood that XOR is a shorthand description for the Boolean logic expression (A & ~B) | (~A & B). Since each of XOR gates 82, 84, 86, 88 has many inputs, the 4-bit output is sensitive to many address bits. A small change in the address can generate a large difference in values of address error check bits AE[3:0]. This characteristic is

ideal for address checking.

[0047] The functions implemented by XOR gates 82, 84, 86, 88 are:

[0048] AE[0] = (A[30] ^ A[26] ^ A[25] ^ A[24] ^ A[23] ^ A[21] ^ A[19] ^ A[18] ^ A[15] ^ A[11] ^ A[10] ^ A[9] ^ A[8] ^ A[6] ^ A[4] ^ A[3] ^ A[0]);

[0049] AE[1] = (A[31] ^ A[30] ^ A[27] ^ A[23] ^ A[22] ^ A[21] ^ A[20] ^ A[18] ^ A[16] ^ A[15] ^ A[12] ^ A[8] ^ A[7] ^ A[6] ^ A[5] ^ A[3] ^ A[1] ^ A[0]);

[0050] AE[2] = (A[31] ^ A[28] ^ A[24] ^ A[23] ^ A[22] ^ A[21] ^ A[19] ^ A[17] ^ A[16] ^ A[13] ^ A[9] ^ A[8] ^ A[7] ^ A[6] ^ A[4] ^ A[2] ^ A[1]);

[0051] AE[3] = (A[29] ^ A[25] ^ A[24] ^ A[23] ^ A[22] ^ A[20] ^ A[18] ^ A[17] ^ A[14] ^ A[10] ^ A[9] ^ A[8] ^ A[7] ^ A[5] ^ A[3] ^ A[2]);

[0052] A simulation of this CRC generator polynomial with 10,000 trials of random address errors and a 32-bit address showed that overall 5.9 % of address errors were undetected. Only 3.8 % of 2-bit errors were not detected, 7.1 % of 3-bit errors were undetected, 6.0 % of 6-bit errors were undetected, and 6.2 % of 5-bit address errors were not detected. Thus about 94% of common multi-bit address errors were detectable with this polynomial. Like

all CRCs, all single-bit errors are detected.

[0053] In addition, the CRC code provides strong protection against so-called "burst" errors. Burst errors are consecutive numbers of wrong bits within the address. Simulation shows that this code detects all burst errors, for 32-bit addresses, up to bursts of length 14.

[0054] Another CRC-generator polynomial is X**4 + X**3 + 1. Simulation with this alternative polynomial produced 6.0 % undetected errors overall, 3.8 % of 2-bit errors undetected, and 7 % of 3-bit errors. All single bit errors are detected. Thus results are similar, although in general not all generator polynomials may be as good.

[0055] When simultaneous address and data errors occur, there may be interference that prevents detection of address errors and/or detection and correction of data errors.

[0056] In contrast, a simulation of 4 address parity bits (parity across 8-bits of address) shows that while all 1-bit errors are detected, 23% of 2-bit errors are undetected and 13.5% of 4-bit errors are undetected. Parity detects all odd-bit errors (1-bit, 3-bit, 5-bit, etc.), but at the expense of even-bit errors. Address multiplexing of DRAMs can produce 2-bit address errors (or any number of even-bit errors), since each address-input pin is used twice for

the row and column addresses. Thus even-bit errors are considered likely. Address parity fails to protect against a type of error that should be common with DRAM memories.

[0057] Parity codes are also very weak in protecting against burst errors, where consecutive bits are wrong. For instance, 90% of burst errors of length 2 are undetected with a 32-bit address with a 4-bit byte parity code. Depending on the arrangement of address bits, this error behavior can happen due to electrical crosstalk or other reasons.

[0058] While parity detects about 91% of random-bit errors, the CRC code detects about 94% of random-bit errors. While this 3% improvement may seem small, the actual improvement in real DRAM memories may be much higher, since parity fails to detect even-bit errors. The CRC code is ideal for detecting such even-bit errors that may be common with multiplexed-address DRAM memories. In addition, there is much superior burst-error coverage with the CRC code. The metric for goodness of a code should match the expected error patterns.

[0059] Another benefit of the CRC code is that the resulting error syndrome can be used to help identify the address bit in error in some cases. Use of 4 address parity bits would

only isolate a single bit error to one of 8 possible address bits.

[0060] If other test or debug information creates a likely suspicion of a single-bit address error, the error syndrome can be used to aid debug of the problem. It is not guaranteed that the presence of such syndrome is caused solely by a single-bit address error. There are 15 unique syndromes caused by single bit address errors, with this CRC used across 32 address bits. The sixteen bit error syndrome is shown, which is a concatenation of zeroes and two copies of the 4 bit error syndromes. Note that any 16 bit error syndrome which has zeroes in bits [15:8] and matching bits in [7:4] and [3:0] can be assumed to be more likely caused by an address error, than an uncorrectable data error.

[0061] Below are examples of error syndromes of merged ECC codeword 30 when a single-bit address error causes the error. The faulty address bits are followed by the syndrome. Only one of the address bits is faulty in each line. If there are more errors in the address than a single bit, or an uncorrectable data error occurs, it is possible to also have these error syndromes. Unlike the correctable data errors, they are not unique error syndromes. This decode

is useful, however, for identifying the source of repeated single bit address errors, when it is known the single bit address errors are more likely than the other causes.

[0062] Addr[0], or Addr[15], or Addr[30]: 0x0033

[0063] Addr[1], or Addr[16], or Addr[31]: 0x0066

[0064] Addr[2], or Addr[17]: 0x00CC

[0065] Addr[3], or Addr[18]: 0x00BB

[0066] Addr[4], or Addr[19]: 0x0055

[0067] Addr[5], or Addr[20]: 0x00AA

[0068] Addr[6], or Addr[21]: 0x0077

[0069] Addr[7], or Addr[22]: 0x00EE

[0070] Addr[8], or Addr[23]: 0x00FF

[0071] Addr[9], or Addr[24]: 0x00DD

[0072] Addr[10], or Addr[25]: 0x0099

[0073] Addr[11], or Addr[26]: 0x0011

[0074] Addr[12], or Addr[27]: 0x0022

[0075] Addr[13], or Addr[28]: 0x0044

[0076] Addr[14], or Addr[29]: 0x0088

[0077] Many other examples could be constructed and other CRC codes could be used.

[0078] ALTERNATE EMBODIMENTS

[0079] Several other embodiments are contemplated by the inventor. For example, other address, data, and ECC widths can be substituted. Not all address or data bits may be checked. Many logical and physical implementations of the functions described herein are possible, with many variations. Some address check or parity bits could be stored and others merged with the data ECC bits. Rather than merge with the data ECC bits, the address check bits could be merged with the data itself. Then the address error is not detected by a repeated error in the lowest 2 nibbles of the ECC codeword. The address error would still be detected by an error syndrome that is not one of the correctable error syndrome patterns. However, the pattern for a "likely" address error syndrome is no longer having the lower two nibbles of syndrome equal to each other, and the upper two nibbles equal to zero. The resulting syndromes could be calculated by generating the expected check bits for the data with such merging, and XORing it with the expected check bits without the merging. This is useful if some info about "likely" address er-

rors is needed. In all cases, the error syndrome will be unique with respect to correctable errors, and be indicated to be uncorrectable.

[0080] Some address bits may not be checked, such as low-order or high-order address bits. The various steps and functions may be pipelined, altering timing. Some address locations may not have ECC storage or may not used a merged ECC codeword while other locations store a merged ECC codeword.

[0081] Other nibbles or bits of the data ECC could be merged with the address check bits rather than the two lowest-order nibbles. The address check bits could be merged with more than two nibbles. Rather than use 4-bit nibbles, other size units could be used. For example, a S5EC/D5ED code could be used, and 5 address check bits could be redundantly merged with the data ECC bits. Alternatively, a S3EC/D3ED code could be used, and 3 address check bits could be redundantly merged with two 3-bit units of the data ECC bits. In general, a SbEC/DbED code could be used, with b address check bits redundantly merged with the two b-bit units of ECC bits, where b is a whole number of at least 2.

[0082] A variety of S4EC/D4ED codes and CRC generator polyno-

mials could be used. Many technical papers have been published exploring and contrasting detection efficiency of different polynomial functions and codes. Codes that have parity matrices in systematic form, or use a rotational construction technique are particularly useful. Some S4EC/D4ED codes may be able to cover more than 128 data bits or less than 128 data bits.

[0083] Functional units could be re-used. For example, A single CRC-code generator could be used for both read and write addresses. A programmable arithmetic-logic-unit (ALU), digital-signal processor (DSP), or other functional unit could be programmed to perform the various operations, or dedicated logic could be used, or some combination. The read and write addresses could share the same physical lines and interface, with a read-write control signal indicating whether the address is a read address or a write address.

[0084] While a CRC code for a 32-bit address has been described, this or another code could be used for larger addresses, such as 40 or 64-bit address, with a degraded error detection capability. Conversely, increased error detection can result if a smaller number of address bits are used (<32).

[0085] CRC codes are desirable for generating the address check bits because the CRC-polynomial requires only 4 check bits. The higher compression of the CRC-polynomial over other codes with similar error detection is an advantage. There is a need to force reporting of all detected address errors as uncorrectable errors in the data SbEC/DbED code. Larger numbers of check bits makes this difficult to achieve For instance, the inventor is unaware of any known S4EC/D4ED codes in systematic form, that cover more than 128 data bits, using only 16 check bits. This means that incorporating 8 address check bits would be difficult.

[0086] Exclusive-NOR (XNOR) gates may be used rather than XOR gates. The invention may be implemented as a memory controller that connects to standard memory modules. The final address check bits, or data check bits, may be complemented, either individually or as a group, before use. The data or address bits may be complemented individually or as a group. A known mechanism for generating CRC codes is to seed a LFSR generator with a non-zero constant. This has the effect of selectively complementing individually bits of the address check bits. A more general function such as a linear block code could be used in

place of the CRC code.

[0087] A (n,k) linear block code is defined by a generator matrix G of dimension n by k, message m of length k, and code c (message plus check bits) of length n, such that c = mG, where modulo-2 arithmetic is used. Each codeword of a linear code is thus some linear combination of the rows of G. The rows of G must be linearly independent. Since systematic linear block codes are used here, the check bits are then n-k bits of c. The CRC code generation method described creates an implied (36,32) linear block code.

[0088] Any advantages and benefits described may not apply to all embodiments of the invention. When the word "means" is recited in a claim element, Applicant intends for the claim element to fall under 35 USC Sect. 112, paragraph 6. Often a label of one or more words precedes the word "means". The word or words preceding the word "means" is a label intended to ease referencing of claims elements and is not intended to convey a structural limitation. Such means-plus-function claims are intended to cover not only the structures described herein for performing the function and their structural equivalents, but also equivalent structures. For example, although a nail and a screw have different structures, they are equivalent structures

since they both perform the function of fastening. Claims that do not use the word "means" are not intended to fall under 35 USC Sect. 112, paragraph 6. Signals are typically electronic signals, but may be optical signals such as can be carried over a fiber optic line.

[0089] The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.